Resource Optmized Runtime (ROR) for Exascale Systems

Dhabaleswar K. (DK) Panda
Department of Computer Science and Engineering, The Ohio State University, Columbus, OH
panda@cse.ohio-state.edu

1 Motivation

The design and development of current generation supercomputing systems is fueled by the increasing use of multi-core processors, accelerators and high-speed interconnects. Current #1 system in the TOP500 list has 1.5 Million cores and delivers 16PF performance with a power budget of 7.9 MW. Exascale systems are expected to have much higher degrees of concurrency, with $O(1K) or O(10K)$ processing units within each node, and $O(100K) or O(1M)$ nodes. Even though these systems may offer significantly higher number of computing cycles, they have a strict power budget of just about 20MW, along with shrinking amounts of caches, physical memory and network bandwidth available per core. Additionally, such systems will also have significantly higher failure rates and smaller Mean Time Between Failures (MTBF). These factors lead us to the following broad challenge: *Can software stacks and runtimes for exascale systems be designed in a highly efficient manner within the limited resource and energy budgets, along with shrinking MTBFs?* We believe that scientific applications cannot achieve true exascale performance on next generation systems unless runtimes are designed from ground-up in a resource optimized manner.

The Message Passing Interface (MPI) has been a popular programming model for scientific parallel applications and has been used successfully to implement regular and iterative codes. Emerging data-driven applications pose new challenges, such as data distribution, load balancing and are harder to address using the MPI paradigm. The Partitoned Global Address Space (PGAS) models present an attractive approach and promise to improve programmability of such applications. Thus, there is an increasing focus on "hybrid" MPI+PGAS programming model which can deliver *both* performance and programmability for exascale systems. This also allows for suitable parts of existing MPI applications to be enhanced using PGAS models. In order to fully leverage the benefits of these diverse and disparate programming models, an ultra-scale, high performance, resilient, unified runtime needs to be designed to work within the shrinking resource envelopes of next generation exascale systems.

2 Vision

We envision a radically new runtime that optimizes resource utilization through fine-grained coordination between processes/ threads that share system components. The runtime should support high performance, scalable and resilient asynchronous communication while meeting the resource constraints of an exascale environment with billions of threads/processes. We propose the design of a Resource Optimized Runtime (ROR) for exascale systems with the following novel features:

Dynamic Multi-Endpoint Connection Management: In order to handle the high communication requirements of an exacale system, ROR must rely on a scalable, light-weight connection management framework that minimizes the memory requirements of establishing network-level connections. Connectionless transports (such as Unreliable Datagram) can be used to achieve this goal, but such an alternative may not offer the best communication performance. Morever, next generation systems will have very high degrees of concurrency with $10^3 - 10^4$ threads within each node. If multiple threads within each node are communicating over the network simultaneously, lock-based mechanisms to ensure consistency will not be scalable. An open challenge is whether ROR can allow multiple threads/processes to seamlessly share a dynamic pool of network connections to deliver high performance, with low memory footprint. Such a runtime can also directly leverage any hardware support offered by next generation network interfaces to manage dynamic connection pools.

Virtual Threading: Each compute node in an exascale system will be based on complex multi-/many-core architectures with non-uniform memory and cache access characteristics. In such architectures, the "Memory Wall" will increasingly dominate the data access latencies and this can significantly affect the performance of parallel applications. We envision ROR to use mutliple virtual threads per hardware core, to address this challenge. When one thread is blocked on local or remote memory access, other threads can occupy the core to do useful computation. In addition, ROR will need to have highly efficient and light weight thread scheduling and locality-aware data/thread placement policies.

Communication Progress via Work Stealing: With increasing number of cores, the limited network resources have to be used efficiently to deliver optimal performance on exascale systems. ROR can achieve this using a multi-threaded design to support fine-grained progress scheduling. In this design, worker threads will be mapped onto OS-level threads to allow threads to easily share and thus, balance the communication workload. Through such a design, idle threads can be used as stand-ins to handle incoming requests on behalf of busy computing threads.

Non-Blocking Collective and Topology Awareness: Non-blocking collective communications are widely expected to offer efficient computation/communication overlap. Such and interface allows applications to hide the communication and synchronization overheads so that they can take full advantage of the high degrees of concurrency offered by exascale systems. Thus,

ROR needs to include highly efficient software or hardware-based non-blocking collective operations. Additionally, the lower level communication substrate needs to automatically detect optimal network paths and route various messages along these paths (considering underlying topology of the network). This will result in efficient use of limited network resources while reducing congestion and increasing overall network throughput.

Out-of-Core Data Management: As the number of cores per node is increasing rapidly, the memory per core and the bandwidth per core is decreasing. PGAS models provide primitives for the potentiality of a convenient and productive mechanism to manage remote memory pools efficiently for out-of-core data. We envision a multi-tiered approach through virtualized cached memory hierarchy to be used in ROR to optimize out-of-core data accesses. Such an approach will automatically manage data movement across these components, including accelerators, to provide a convenient and efficient mechanism to support out-of-core algorithms.

Power-Aware Designs: Numeruous opportunities exist for conserving power across various system components: host processors, memory, accelerators, network fabric and storage devices. Thus, it will be critical for ROR to include power-saving schemes in all mechanisms and services, so that applications can leverage ROR to maximize power savings without any sacrifice in performance.

3 Related Work

MPI [1] has been the most widely used parallel programming model in the HPC domain and some highly optimized MPI runtimes [2, 3, 4] are available, scaling it to multi-petaflop systems. Several studies have highlighted the limitations that these runtimes face in terms of resources as we move towards exascale computing [5, 6]. PGAS models have gained popularity in the recent years, promising better programmability and performance than MPI for applications with dynamic communication patterns [7, 8, 9, 10]. GASNet is a networking layer that has been widely used to build runtimes for PGAS languages and libraries [11]. Though GASNet has been widely ported to modern platforms, it lags behind in performance, scalability and features when compared to popular MPI runtimes. There has been wide support for hybrid programming models. Runtime designs which can support both MPI and PGAS models have been proposed [12]. A runtime designed and developed at OSU now supports concurrent use of UPC, OpenSHMEM, and MPI and delivers high performance for the differnt combinations of these models [13, 14]. Our proposed ROR runtime builds on the experience of designing this unified runtime and promises designs to address the resource challenges that runtimes face at exascale.

Ideas like MPI endpoints and allowing mutliple MPI processes to share an address space are being actively discussed in the MPI community [15]. They are expected to improve resource sharing and provide better support for hybrid programming models. An efficient runtime support is required for these models to deliver the expected benefits [16]. MPI-3, which is soon to be standardized, has added features like non-blocking collectives and enhanced one-sided communication to allow for better resource utilization through overlap. MPI runtime developers are exploring both host-based [17] and network-based solutions [18, 19] to provide efficient support for these features. Exploring topology-aware process placement and communication strategies in MPI is also an area of active research [20, 21].

4 Assessment

Challenges Addressed: The fundamental issues in designing high performance and scalable runtime for exascale systems with constraints of various resources (memory, cache and network bandwidth per core and power) are addressed here. The challenges include latency hiding, minimizing data movemement, energy-saving, and limited I/O capabilities.

Maturity: Several groups in the HPC community, including the PI's group, have initiated research along the proposed direction. Some of the initial studies, done along these directions, are outlined in Section 3. The results of these studies and the associated runtimes have enabled the design and deployment of multi-Petaflop systems during the last few years. However, significant research and development is needed (as discussed in Section 2) to design the proposed ROR for exascale systems with a billion processes/threads.

Uniqueness: The resource constraints (memory, cache and network bandwidth per core and power), indicated in this position paper, are unique to exascale systems. Thus, the proposed ROR design and the associated approaches are unique for exascale systems and will not be addressed by other programs.

Novelty: The proposed ROR design focuses on a ground-up approach with a billon processes/threads and the associated resource constraints in mind. Thus, it has novel designs which are not available in existing solutions.

Applicability: Exascale systems are being targeted for not only high-end computing, but also for Big Data processing and analytics. The proposed ROR will be applicable to both kinds of exascale systems.

Effort: A multi-year (3-4 years) multi-institutional effort is needed to design the proposed ROR runtime, and validate its effectivness with emerging exascale systems and applications.

References

[1] *MPI: A Message-Passing Interface Standard*, Message Passing Interface Forum, Mar 1994.

[2] "MPICH2," http://www.mcs.anl.gov/research/projects/mpich2/.

[3] "MVAPICH: MPI over InfiniBand, 10GigE/iWARP and RoCE," http://mvapich.cse.ohio-state.edu/.

[4] "OpenMPI," http://www.open-mpi.org.

[5] P. Balaji, D. Buntinas, D. Goodell, W. Gropp, S. Kumar, E. Lusk, R. Thakur, and J. L. Träff, "MPI on a Million Processors," in *Proceedings of the 16th European PVM/MPI Users' Group Meeting on Recent Advances in Parallel Virtual Machine and Message Passing Interface*, 2009.

[6] D. Goodell, W. Gropp, X. Zhao, and R. Thakur, "Scalable memory use in MPI: a case study with MPICH2," in *Proceedings of the 18th European MPI Users' Group Conference on Recent Advances in the Message Passing Interface (EuroMPI)*, 2011.

[7] "Berkeley Unified Parallel C (UPC) Project," http://upc.lbl.gov/.

[8] "Co-Array Fortran," http://www.co-array.org/.

[9] "The X10 Programming Language," http://domino.research.ibm.com/comm/research_projects.nsf/pages/x10.index.html.

[10] "OpenSHMEM Application Programming Interface," http://www.openshmem.org.

[11] "GASNet: Global-Address Space Networking," http://gasnet.cs.berkeley.edu/.

[12] D. Buntinas and W. Gropp, "Designing a Common Communication Subsystem," in *Euro PVM/MPI 2005 Conference, September 2005*, 2005.

[13] J. Jose, M. Luo, S. Sur, and D. K. Panda, "Unifying UPC and MPI Runtimes: Experience with MVAPICH," in *Fourth Conference on Partitioned Global Address Space Programming Model (PGAS)*, 2010.

[14] J. Jose, K. Kandalla, M. Luo, and D. K. Panda, "Supporting Hybrid MPI and OpenSHMEM over InfiniBand: Design and Performance Evaluation," in *Int'l Conference on Parallel Processing (ICPP '12) (Accepted for publication)*, 2012.

[15] "MPI Forum: Hybrid Programming Working Group," https://svn.mpi-forum.org/trac/mpi-forum-web/wiki/MPI3Hybrid.

[16] M. Luo, J. Jose, S. Sur, and D. K. Panda, "Multi-threaded UPC Runtime with Network Endpoints: Design Alternatives and Evaluation on Multi-core Architectures," in *Int'l Conference on High Performance Computing (HiPC '11)*, Dec. 2011.

[17] T. Hoefler and A. Lumsdaine, "Optimizing non-blocking Collective Operations for InfiniBand," in *Proceedings of the 22nd IEEE International Parallel & Distributed Processing Symposium, CAC'08 Workshop*, Apr. 2008.

[18] K. Kandalla, U. Yang, J. Keasler, T. Kolev, A. Moody, H. Subramoni, K. Tomko, J. Vienne, and D. K. Panda, "Designing Non-blocking Allreduce with Collective Offload on InfiniBand Clusters: A Case Study with Conjugate Gradient Solvers," in *IEEE International Symposium on Parallel and Distributed Processing (IPDPS)*, 2012.

[19] R. Graham and S. Poole and P. Shamis and G. Bloch and N. Bocha and H. Chapman and M. Kagan and A. Shahar and I. Rabinovitz and G. Shainer, "Overlapping Computation and Communication: Barrier Algorithms and Connectx-2 CORE-Direct Capabilities ," in *Proceedings of the 24th IEEE International Parallel and Distributed Processing Symposium, Workshops*, 2010.

[20] H. Subramoni, S. Potluri, K. Kandalla, B. Barth, J. Vienne, J. Keasler, K. Tomko, K. Schulz, A.Moody, and D. K. Panda, "Design of a Scalable InfiniBand Topology Service to Enable Network-Topology-Aware Placement of Processes," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC) (Accepted for publication)*, 2012.

[21] A. Bhatele, "Automating Topology Aware Mapping for Supercomputers," Ph.D. dissertation, Dept. of Computer Science, University of Illinois, August 2010.